intel®

APPLICATION NOTE

AP-363

September 1992

3

Extended Flash Bios Design For Portable Computers

SALIM FEDEL
SENIOR APPLICATIONS ENGINEER

Order Number: 292098-002

EXTENDED FLASH BIOS DESIGN FOR PORTABLE COMPUTERS

CONTENTS

CONTENTS

*Microsoft is a registered trademark of Microsoft Corporation.

1.0 INTRODUCTION

PC BIOS has been migrating to Flash-based designs with the introduction of highly optimized Flash memory architectures. The first phase of this shift in paradigm was from ROM/EPROM-based BIOS to Bulk Erase Flash memory-based BIOS to provide for in-system updatable BIOS and hence an easy update capability when BIOS changes are required.

The second phase improved the basic flash design to migrate towards boot block flash memory architecture with the Intel 28F001BX Flash memory. This improvement enabled the implementation of additional features and provided a true design capability for portable PC BIOS.

The third phase of this paradigm shift now starting to evolve, deals with the need to grow beyond the traditional BIOS space limit of 128 KBytes imposed by the original PC architecture to accommodate the advanced features of today's portable and desktop systems.

This application note describes in detail this third phase in BIOS hardware and software implementation. Specifically it will investigate why BIOS needs to grow beyond the 128 KByte code size. Then, a design example using the Intel 28F200BX Boot Block Flash memory in an Intel386SL™-based portable system will be explained in terms of both hardware and software viewpoints. Finally, low voltage PC BIOS designs incorporating 3.3V components are described.

2.0 PC BIOS TODAY AT THE 128 KBYTE CODE SIZE LIMIT

The Basic Input/Output System (BIOS) code is the lowest system level software which manages the interaction between all hardware components (CPU, Chip-Sets and I/O) with all software modules (Operating Systems and Applications Code). BIOS manages many functions in a PC, such as Power-On Self Test (POST), input vector creation, I/O services and system initialization. Therefore BIOS is the essential interface layer for full system functionality and compatibility.

The original PC architecture, developed in 1981, put restrictions on the size and mapping of the BIOS code which was then a very simple piece of software (on the order of 32 KBytes for the original BIOS code). It was located at the top of the PC's (8088) memory map which at the time was a maximum of 1 MByte.

Then in 1984, the PC AT (80286) BIOS was expanded another 32 KBytes for a total of 64 KBytes. Subsequently, towards the late 1980s, more elaborate BIOS set-up utilities started to be an integral part of the BIOS code. In addition, personal computer manufacturers designed custom features into their BIOS code to offer more system flexibility. This increase in code complexity expanded the AT BIOS code another 64 KBytes (for a total of 128 KBytes) to occupy the total BIOS reserved space in the 1 MByte memory map.

In the DOS memory map, BIOS is mapped down from the top of the 1 MByte address space (F0000H to FFFFFH). Additional BIOS code space is available for future enhancements from E0000H to EFFFFH. The next 256 KBytes in the DOS memory map are reserved for adapter space to accommodate add-in boards (for enhanced graphics cards for instance). Finally the remaining 640 KBytes are reserved for the user to load his/her applications for execution. See Figure 1 for a graphical description.

3.0 WHY BIOS CODE WILL GROW BEYOND 128 KBYTES

As advances in computer design affect both desktop systems (with the addition of EISA and SCSI capabilities) and portable systems (with the addition of advanced power management capability and I/O cards), the need for larger amounts of non-volatile memory space becomes evident.

A Notebook or a Palmtop computer design, for instance, may put the operating system, the system management code, set-up or utility programs into the non-volatile memory area to conserve precious RAM space for applications.

Additionally, Video BIOS can also be mapped into the Flash BIOS area as in the case of the Intel386SL Microprocessor Superset.

Therefore, to implement advanced capabilities and provide new features (as described above) into powerful mobile computers, the 128 KByte BIOS code size limit had to be removed as it shall be explored in the next section.

The BIOS of today and the future must adapt to the new requirements of portable PC designs and take advantage of the new capabilities of low power PC chipsets and I/O devices to achieve the highest performance and longest battery life at the lowest system cost.

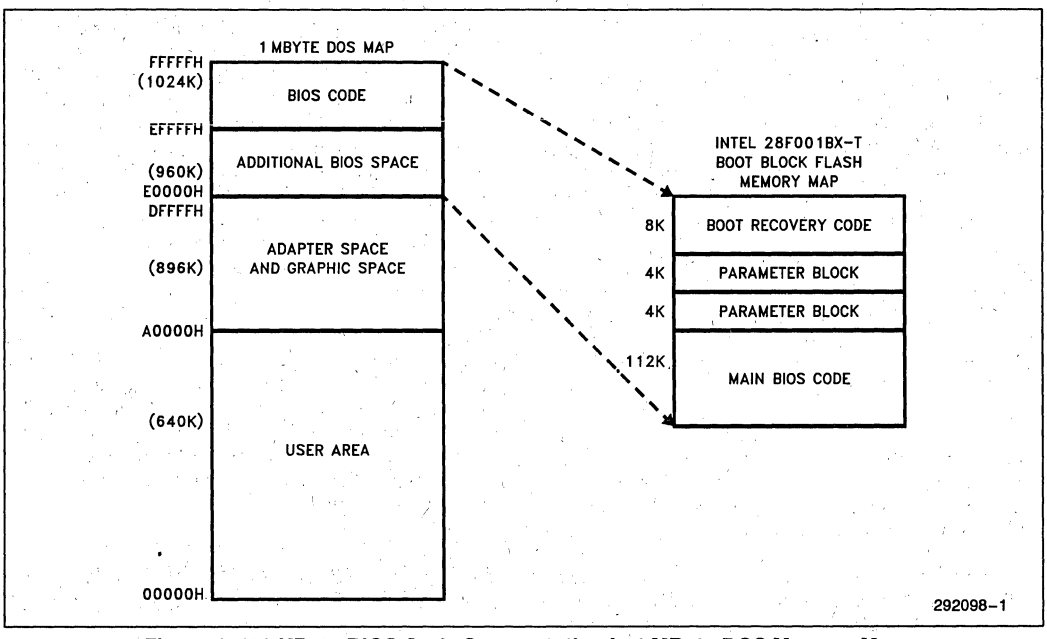


Figure 1. 128 KByte BIOS Code Segmentation in 1 MByte DOS Memory Map

3.1 Advanced Power Management

High integration CPUs and chip-sets such as the Intel386SL Microprocessor SuperSet allow for the design of light, small form factor portable computers with long battery life.

BIOS is the ideal place for implementing the new power management techniques available with the Intel386SL Microprocessor Superset. BIOS software vendors have implemented APM code for the latest generation of Notebook PCs.

This added level of functionality imposed on the BIOS code increases the need for larger code space beyond the traditional 128 KByte BIOS implementations seen in today's portable systems.

APM code typically requires an additional 32 KByte of code space beyond the basic 64 KByte standard BIOS. Therefore, with the addition of APM, BIOS code grows to 96 KBytes.

3.2 Optimize New Portable Applications

With the implementation of I/O card capabilities for non-volatile file storage (with flash memory cards) and the ability to communicate over a telephone line (with modem cards), mobile computers have truly become powerful tools on the road.

The establishment of a common PC card standard for full system compatibility is described in the PCMCIA Standard release 2.0 (Personal Computer Memory Card International Association). Intel has developed a similar set of specifications fully compatible with the PCMCIA release 2.0 standard called the Exchangeable Card Architecture (ExCA). In order to implement ExCA capability in a portable computer, additional BIOS code called Socket Services is required to manage the system I/O card functionality. To implement the ExCA specifications, socket services needs an additional 16 KByte of code space.

Furthermore, in the pen-based PC applications, there are even greater BIOS requirements to design-in unique features such as: pen extensions, touchscreen capability and character recognition interface code. These new features require the implementation of additional BIOS code (may be 16 KBytes to 32 KBytes).

To take full advantage of Desktop system capabilities and performance while still having a portable computer to take on the road, docking station designs were conceived. This added level of complexity for the portable computer increases the code required in a basic system BIOS.

3.3 Putting Microsoft MS-DOS 5.0 Operating System ROM Version into the BIOS Chip

Additionally, MS-DOS 5, ROM version is now becoming a standard in virtually all diskless sub-notebook, notebook and pen PC implementations. Many factors contribute to this approach. Chief among them is the reduced disk access and the resulting longer battery life. Another factor is the instant boot capability which is essential in certain applications.

Today's MS-DOS 5, ROM version occupies 64 KBytes of code space as specified from the Microsoft Product Description.

3.4 Relocated Resident VGA Code: VIDEO BIOS

As described above in section 3.0, video BIOS can also be mapped into the system Flash BIOS memory allowing the entire system non-volatile storage requirements to be satisfied with one Flash device. Resident VGA BIOS code takes approximately another 32 KBytes of memory space.

In summary, adding all the above code size requirements, the resulting BIOS storage area increases from the initial 128 KByte requirement to somewhere between 208 KBytes (without pen extensions) to 240 KBytes (including a pen input capability).

There are already portable designs today which have filled a 256 KByte code space to accommodate some of the above mentioned needs.

4.0 HARDWARE DESIGN FOR A 256 KBYTE BIOS

4.1 Intel 28F200BX/002BX Boot Block Flash Memory Family

Building upon the wide acceptance of the Intel28F001BX 1 Mbit flash memory for BIOS designs, a new family of higher density flash components is now available to solve the PC designer's need of implementing extended BIOS code beyond 128 KBytes.

These new flash memories at the 2 Megabit density levels, are structured around the same boot block architecture as the Intel 28F001BX and are therefore compatible. They provide block erasure capability, boot code hardware protection and very low power consumption as in the case of the 28F001BX.

In addition, they incorporate new features to simplify the device interface and allow the system designer to optimize platform designs.

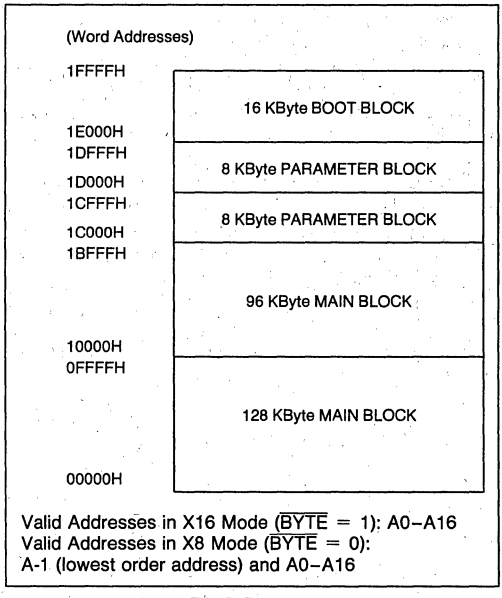
These new features are summarized as follows:

- User selectable 8-bit or 16-bit read/write operation (28F200BX)
- 60 ns access time performance
- 16 KBytes Boot Block space which is hardware protected
- Two, 8 KBytes Parameter Blocks
- One, 96 KByte Main Block
  One, 128 KByte Main Block
- 8-bit only operation and packaging for space sensitive applications (28F002BX)

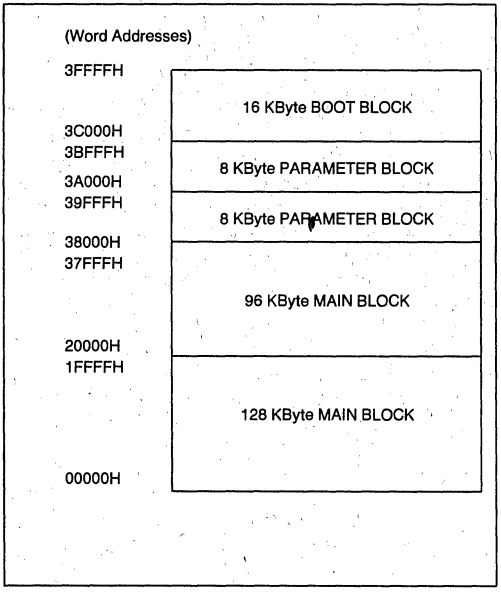
In this section, we will describe these new features in more detail and discuss their system applicability.

The blocking scheme, while still of the boot block type, is expanded by defining additional blocks (2 main blocks) to allow for software modularization and a self-contained design.

As the size of the BIOS code stored in any one device grows due to the complexity and high integration of chip-sets, so does the "kernel" code stored in the boot section. The boot and parameter blocks were accordingly doubled in size in comparison to the 28F001BX device. The two parameter blocks of 8 KBytes each allow the PC designer to store BIOS extensions or Battery-Backed SRAM configuration data (CMOS RAM, EISA configuration parameters). The two main blocks are used to store the main BIOS code in modular fashion if so desired for future easy updates. These main blocks can also be used to store ROM-executable Operating System software such as MS-DOS 5, ROM version or drivers and utilities. Refer to Figure 2 for the block locations for both the 28F200BX-T and 28F002BX-T.



28F200BX-T Top Boot Map



28F002BX-T Top Boot Map

Figure 2. 28F200BX-T/002BX-T Memory Maps

In addition, the 28F200BX/002BX devices incorporate new capabilities desired by today's sophisticated PC designer.

The byte-wide or word-wide feature available as a designer-selectable option gives the ability to interface to an 8-bit or 16-bit wide bus. The performance and hardware goals of some systems may require 16-bit BIOS data bus. A system with a small amount of RAM and a large amount of flash memory-based operating system code for example, may require a 16-bit BIOS data bus to maintain a high performance level of operation.

The high access speed of these new devices, which for the first time break the 60 ns barrier, is another big advantage the PC designer can fully exploit to increase system performance and acceptance in the marketplace. For example, a PC designer may choose to execute BIOS directly out of flash memory instead of shadowing to system RAM as well as execute MS-DOS 5, ROM version out of flash for instant-on capability and to achieve better overall system performance.

Block erasure allows independent modification of code and data and maximum flexibilty in production as well as after the system is shipped. The boot block, which is hardware protected, insures that minimal BIOS code is present to always boot up the system successfully. The 16 KByte boot block is protected from alteration during system power excursions by a high voltage pin. This write protection pin called $\overline{PWD}$ has to transition to 12V with the normal Vpp voltage pin to allow for boot block write and erase operations.

If systems are designed with the ability to switch $\overline{PWD}$ to high voltage (12V), guaranteed full non-volatility of the boot code is achieved. This feature always guarantees system recovery from power failure and provides the security needed for the end user when performing BIOS code updates.

To meet the crucial needs of lower power consumption, the 28F200BX/002BX devices incorporate a deep-power down current mode activated through the $\overline{PWD}$ pin under the TTL/CMOS level control. When this pin transitions to ground the device typically consumes 1 microwatt through the V_{CC} supply pin.

In addition, the 28F200BX/002BX devices include an Automatic Power Savings feature during active mode of operation. This feature allows the memory chip to put itself in a very low current state when it is enabled but not accessing a new memory location.

The 28F200BX/002BX devices incorporate an internal Write State Machine, Command User Interface and a Status Register to fully control the program and erase operations and greatly simplify the user write and erase algorithms and hence the update code procedure. They also include an erase suspend feature which allow the system to service interrupts and access the device during BIOS code updates (refer to appendix B).

Finally, the 28F200BX/002BX, 2 Megabit devices have an equivalent 4 Megabit boot block flash memory family of devices called the 28F400BX/004BX, allowing for easy density upgrade and total compatibility between systems using both types of memories. Refer to the documentation mentioned in the reference section of this application note.

Figure 3 is a block diagram description of the 28F200BX/002BX products.

4.2 Intel386SL™ Microprocessor Superset Platform Overview

3

This design example focuses primarily on how to interface the Intel 28F200BX-T or 28F002BX-T Boot Block Flash memories to the Intel386SL Microprocessor Superset in a 16-bit wide or an 8-bit wide configuration respectively. This is an extended BIOS design example which demonstrates how the barrier of the 128 KBytes BIOS size memory is eliminated. Figure 4 shows an interface diagram of the Intel 28F200BX-T Boot Block Flash memory (128 K x 16) to the Intel386SL Microprocessor superset. Figure 5 shows an equivalent interface diagram of the Intel 28F002BX-T Boot Block flash memory (256 K x 8) to the Intel386SL Microprocessor Superset.

The Intel386SL Microprocessor Superset Flash BIOS interface supports up to 256 KBytes of Flash BIOS (a 2 MEGABIT Flash memory device) to enable the system designer to meet specific design goals as described in section 3 above.

The Intel386SL Microprocessor Superset supports the following features:

- Up to 256 KBytes Flash BIOS
- VGA BIOS mapping into system BIOS
- 8-bit or 16-bit BIOS interface
- Programmable number of flash wait states for read access (from zero to fifteen Wait-States to optimize the system performance)
- BIOS shadowing mechanism

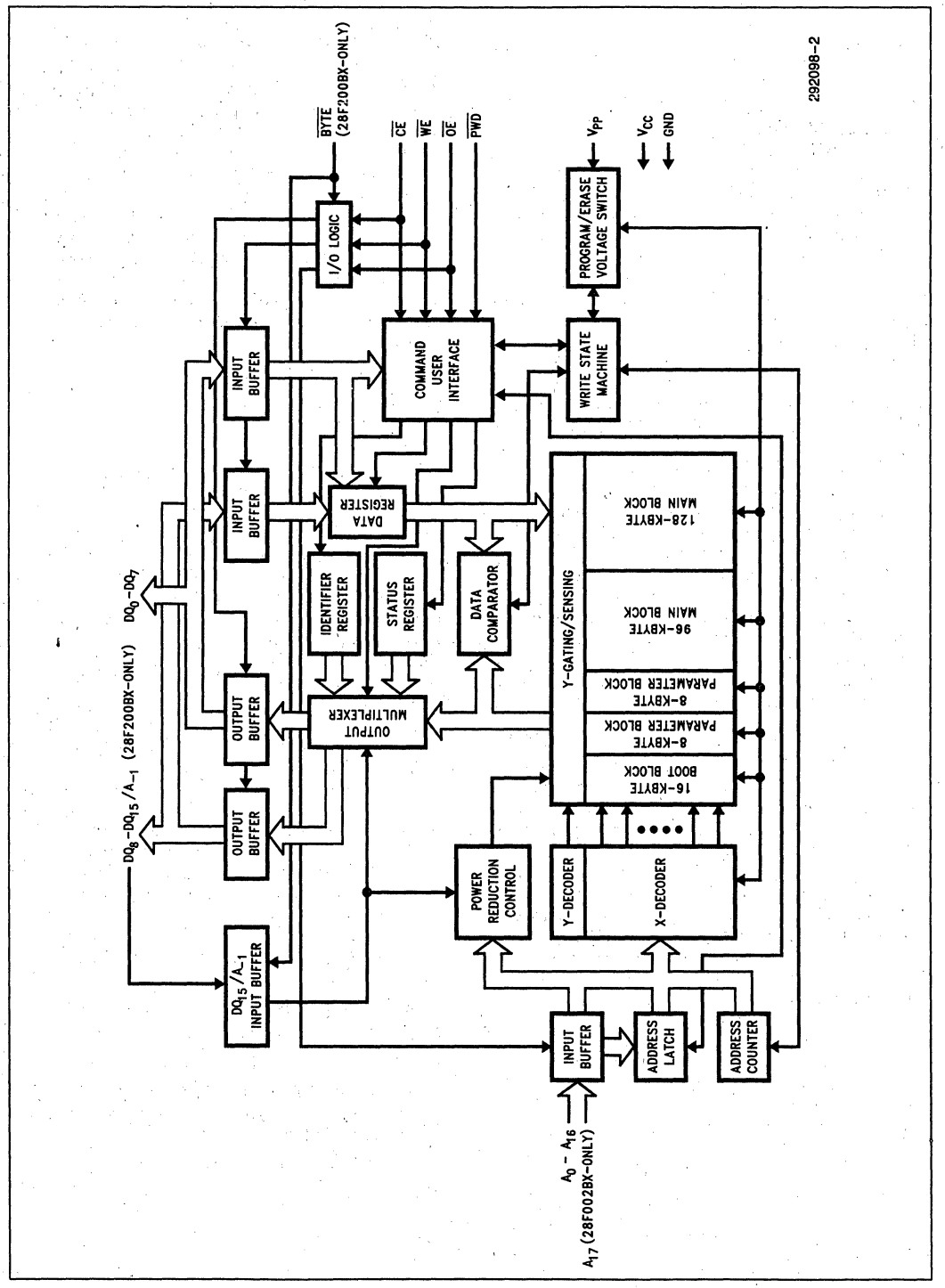


Figure 3. Block Diagram of the 28F200BX/28F002BX

Flash BIOS size configurations in the Intel386SL Microprocessor Superset system are controlled by programming certain registers located in the normal I/O address space, namely:

- EBCICR: External Bus Unit Configuration Register 1 located at 300H
- ROMCS_DEC: ROM Chip Select Decode register located at Index 2FH
- ISAWINDOW: ISA Window control register located at B00H

When a 256 KByte Flash BIOS configuration is programmed into the Intel386SL Microprocessor Superset, 128 KBytes are directly accessible in the E0000H–FFFFFH address range. The other 128 KBytes are decoded at the top of the 16th or 32nd Megabyte of the Intel386SL superset address space, i.e., either:

FC0000H–FDFFFFH or 1FC0000H–1FDFFFFH. This extra ROM space is accessed by programming the ISA sliding control register to point to one of these two areas and then accessing the ISA sliding window in the D0000H–DFFFFH address range (64 KBytes). This mechanism allows complete access of the 256 KBytes of BIOS code without having to enter the Intel386SL protected mode.

The BIOS code size is used to internally decode the ROM address space and generate two chip select signals: $\overline{\text{ROMCS0}}$ and $\overline{\text{ROMCS1}}$, to control the Flash BIOS device. In the case of a single Flash BIOS device, only $\overline{\text{ROMCS0}}$ is needed to drive the $\overline{\text{CE}}$ chip select signal of the flash memory.

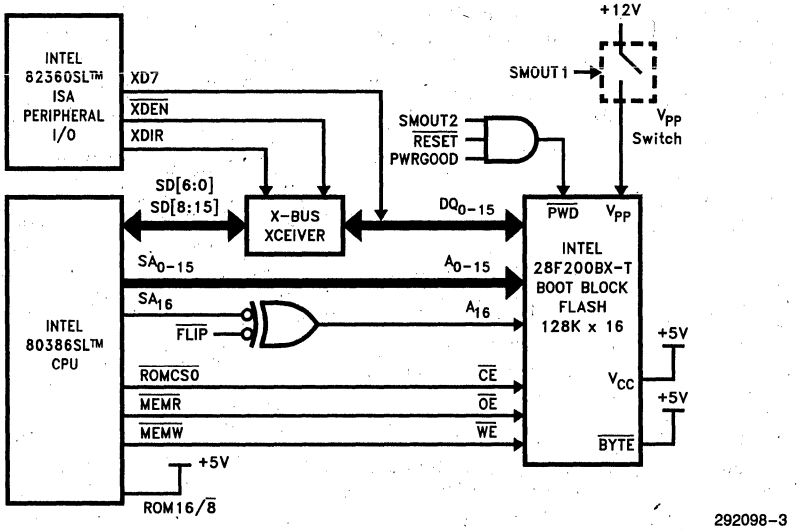
In the diagram of Figure 4 note the following:

- $\overline{\text{BYTE}}$ signal is set high to enable 16-bit operation of the flash device.
- The highest order system address line SA_{16} is inverted when $\overline{\text{FLIP}}$ signal becomes active at boot-up time to relocate the boot kernel code at the top of the 1 MByte memory map for the system to boot from it. (See section 4.5).
- ROM16/$\overline{8}$ is set high to enable 16-bit bus operations.
- $\overline{\text{PWD}}$ signal is gated by the PWRGOOD signal (for reset when power fails) and by system RESET signal.
- V_{PP} supply voltage is switched to the flash device only when BIOS updates are required.

In addition to the above considerations, note the following in Figure 5:

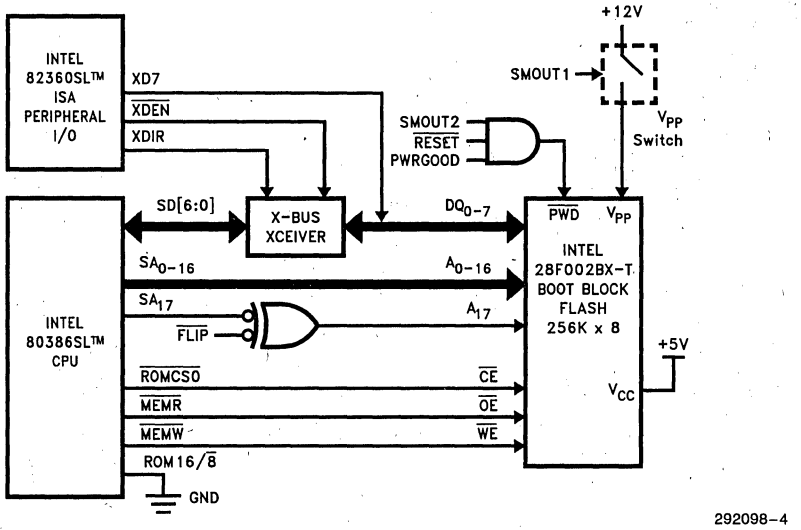
- The highest order system address line SA_{17} is also inverted when $\overline{\text{FLIP}}$ signal becomes active at system boot-up time.
- ROM16/$\overline{8}$ is set low to enable 8-bit only bus operations.

3



LEGEND:
XDEN = X-BUS DATA ENABLE
XDIR = X-BUS DATA DIRECTION
SMOUT1,2 = SYSTEM MANAGEMENT OUTPUT CONTROLS
ROMCS0 = ROM CHIP SELECT FOR SYSTEM FLASH BIOS
MEMR = MEMORY READ
MEMW = MEMORY WRITE
FLIP = BOOT BLOCK MEMORY MAPPING SIGNAL
ROM 16/8 = ROM 16 BITS OR 8 BITS
SD[6:0], SD[15:8] = SYSTEM DATA BUS
SA0–16 = SYSTEM ADDRESS BUS
XD7 = X-BUS DATA BIT 7
PWRGOOD = POWER SUPPLY POWER GOOD SIGNAL

Figure 4. Intel386SL™ Microprocessor Superset with the 28F200BX-T Flash BIOS Chip



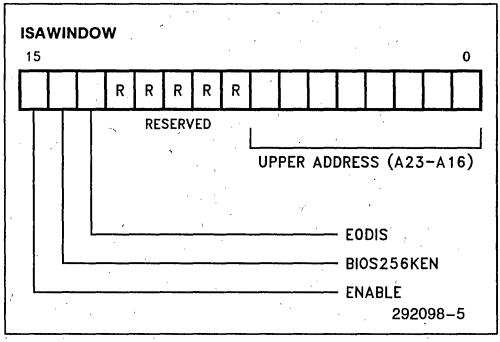
LEGEND:
XDEN = X-BUS DATA ENABLE
XDIR = X-BUS DATA DIRECTION
SMOUT1,2 = SYSTEM MANAGEMENT OUTPUT CONTROLS
ROMCS0 = ROM CHIP SELECT FOR SYSTEM FLASH BIOS
MEMR = MEMORY READ
MEMW = MEMORY WRITE
FLIP = BOOT BLOCK MEMORY MAPPING SIGNAL
ROM 16/8 = ROM 16 BITS OR 8 BITS
SD[6:0] = SYSTEM DATA BUS
SA0–17 = SYSTEM ADDRESS BUS
XD7 = X-BUS DATA BIT 7
PWRGOOD = POWER SUPPLY POWER GOOD SIGNAL

Figure 5. Intel386SL™ Microprocessor Superset with the 28F002BX-T Flash BIOS Chip

4.3 Programming the Intel386SL™ Registers

ISA SLIDING WINDOW PROGRAMMING: ISAWINDOW

When the ISA window is enabled, any access to the area of memory from D0000H to DFFFFH is remapped according to the upper address field specified in the ISAWINDOW control register. To use the ISA sliding window mechanism, the ISAWINDOW enable bit Bit 15 of the ISAWINDOW control register must be set. See the diagram below for a definition of the ISAWINDOW register:



Bits 0–7: ISA Window upper Address bits

Corresponds to the upper address bits of the remapped address, A23–A16

Bit 7 = A23, Bit 0 = A16

Bit 8–12: Reserved. Should be reset to zero for proper system operation

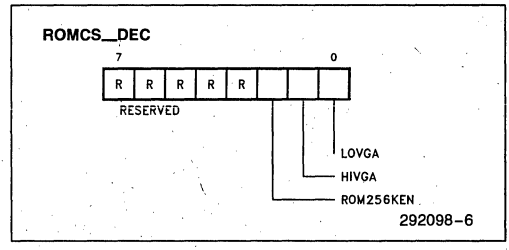
Bit 13: E0000H segment disable.
Normal ISA-bus cycles are generated for access to address range E0000H–EFFFFH.

Bit 14: BIOS256KEN, 256 KByte Flash enable. Enables support for a 256 KByte Flash device.

Bit 15: ISA-Window Enable.

ROM CHIP-SELECT DECODE DEFINITION: ROMCS_DEC

The ROM chip-select decode register is used to indicate to the Intel 82360SL I/O chip how to control the X-bus buffers and XD7 signal with regard to BIOS accesses. The Intel386SL Superset allows systems to have VGA BIOS to physically reside in system BIOS code space. See the diagram below for a definition of the ROM chip-select register:



Bit 0: LOVGA

Low VGA BIOS mapped into system BIOS.

Bit 1: HIVGA

High VGA BIOS mapped into system BIOS.

Bit 2: ROM256KEN

256 KByte ROM enable. When this bit is set, the Intel 82360SL will properly assert $\overline{XDEN}$ and route XD7 through SD7 properly for accesses to the following ranges:

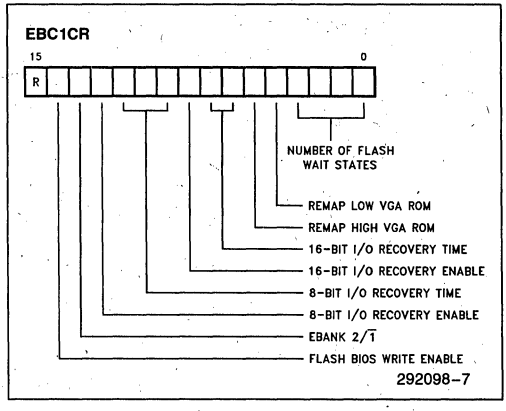
E0000H–FFFFFH

FC0000H–FFFFFFH

Bits 2–7: Reserved

EXTERNAL BUS UNIT CONFIGURATION REGISTER 1:EBC1CR

This register has many functions. It controls the number of flash memory wait states, the mapping of flash memory chip-selects, the I/O recovery time for the ISA-bus and write cycles to the Flash device. The following diagram depicts the definition of this register.



Bit 0–3: Number of Flash wait states.

Flash access cycles with zero to fifteen wait states can be programmed i.e.

[0,0,0,0] represents 0 wait states.

[1,1,1,1] represents 15 wait states.

Bit 4: Low VGA ROM remap

Allows VGA BIOS to be remapped into system BIOS Flash from E0000H to E3FFFH.

Bit 5: High VGA ROM remap

Allows VGA BIOS to be remapped into system BIOS Flash from E4000H to E7FFFH.

Bits 6–7: 16-bit I/O recovery time

Bit 8: 16-bit I/O recovery enable

Bits 9–11: 8-bit I/O recovery time

Bit 12: 8-bit I/O recovery enable

Bit 13: EBANK2/$\overline{1}$, Number of BIOS Flash Banks, 2 or 1

Used to determine whether one (low) or two (high) banks of BIOS Flash are used in the system.

Bit 14: Flash BIOS write enable

Used to enable writes to the Flash device. When this bit is set, write accesses can occur and BIOS updates are possible.

Bit 15: Reserved

4.4 The ISA Sliding Window

The ISA sliding window mechanism is similar to the Expanded Memory System or EMS mechanism. It allows the access of extended memory on the ISA bus in real mode. Hence, there is no conflict in memory allocation when the BIOS chip is accessed during system power-up. This is a straightforward method of implementing a 256 KByte extended Flash BIOS in an Intel386SL-based Personal Computer. Figure 6 shows how extended memory is accessed with the ISA sliding window.

Therefore to use a 256 KByte Flash device in an Intel386SL Microprocessor Superset design, the following steps are specified:

Example:

Enable support of 256 KByte Flash BIOS

Enable Flash BIOS write access

One BIOS Flash Bank used

Program zero Flash wait states (for 16 MHz Intel386SL with a 60 ns Intel 28F200BX-T or 28F002BX-T).

Program the ISA Sliding Window to access memory between FC0000H and FDFFFFH.

if external bus unit configuration space is not enabled

enable external bus unit configuration space
set bit 13 to one
set bit 14 to one
set bits 0–3 to [0,0,0,0]

if ROM Chip-select decode register is not enabled

enable ROM chip-select decode register
set bit 2 to one

enable ISAWINDOW control register

set bit 15 to one
set bit 14 to one

set upper address field of the ISAWINDOW register to FCH

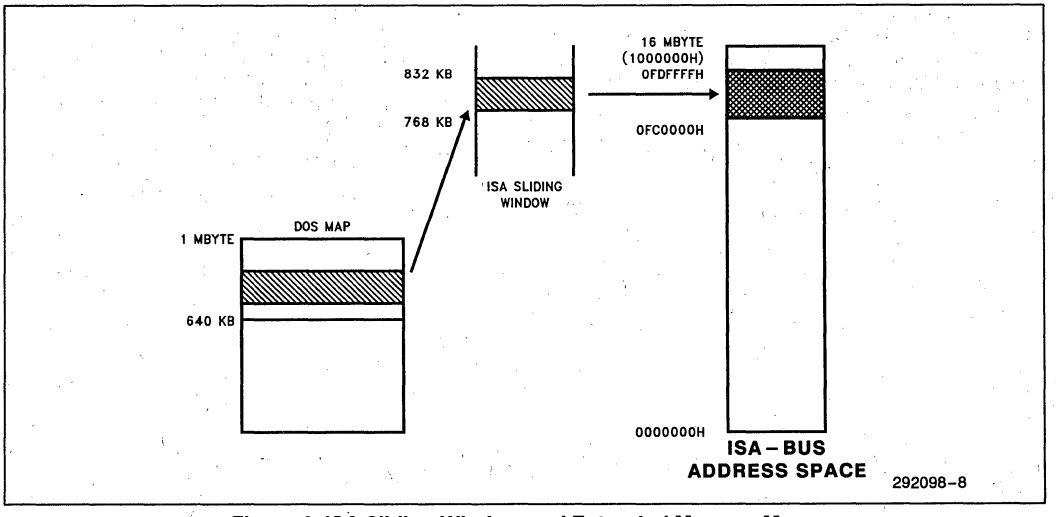


Figure 6. ISA Sliding Window and Extended Memory Maps

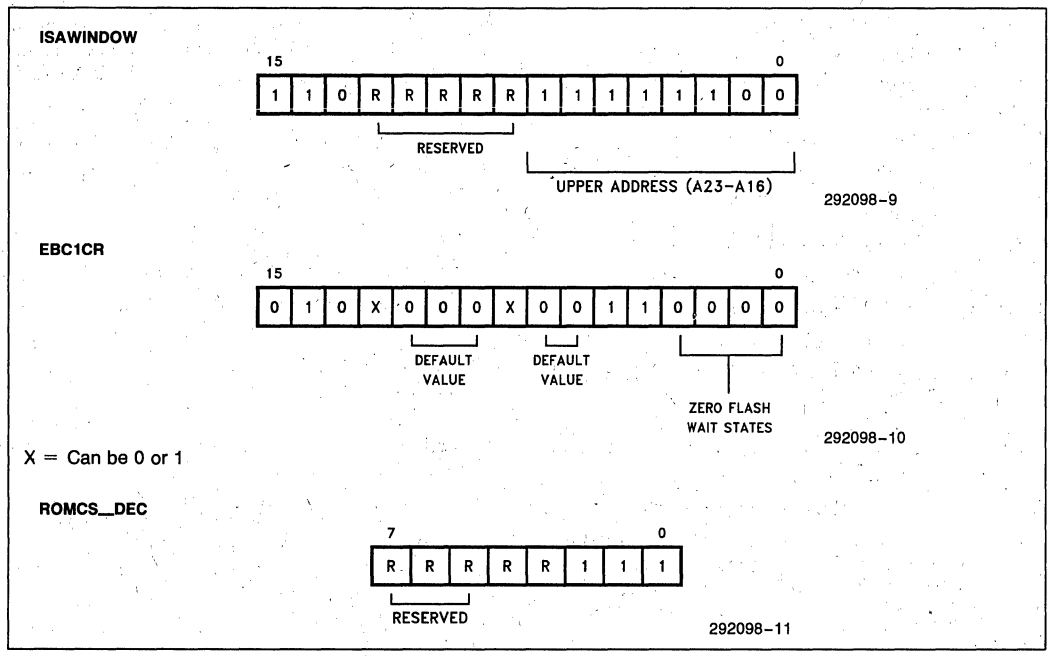


Figure 7. Programming of the Intel386SL™ Microprocessor Superset Registers

4.5 The 28F200BX-T/28F002BX-T in the 1 MByte DOS Memory Map

In addition, due to the above mapping considerations when interfacing a 256 KByte Flash BIOS chip to the Intel386SL™ Microprocessor Superset, it is necessary to flip the 256 KByte Flash device in half at boot time to relocate the Boot Block in the correct physical address at FFFFFH for the Intel386SL™ CPU to boot from it. This is due to the fact that the Intel386SL™ Microprocessor Superset uses the bottom 128 KBytes of the 256 KByte Flash chip to map down to the real mode 128 KByte allocated BIOS space.

Figure 8 shows how the Intel 28F200BX-T flash device fits in the DOS 1 MByte memory map. The same memory map applies to the 28F002BX-T device when designing an 8-bit only system.

5.0 SOFTWARE DESIGN CONSIDERATIONS

The subject of BIOS code update is already discussed extensively in references [1] and [3]. Appendix A of this application note is an example of a flash BIOS update routine. In this section, we mainly discuss considerations of code segmentation and describe how the system handles the different pieces of code under various conditions.

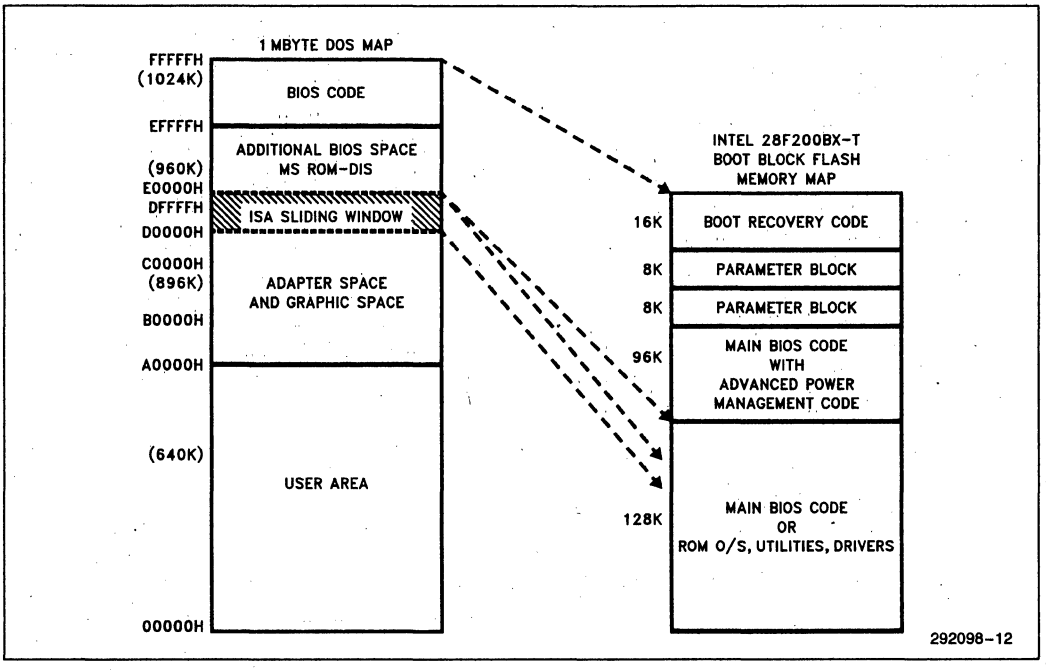


Figure 8. 28F200BX-T/28F002BX-T in the 1 MByte DOS Memory Map

3

As explained in section 3, advanced notebook designs require a large ROM space to conserve valuable RAM space for applications. In addition to the boot kernel code and standard BIOS code, one may put the following code modules:

- APM code
- VGA BIOS
- ExCA socket services
- Operating Systems such as Microsoft MS-DOS 5.0 ROM version

When designing a 256 KByte BIOS system as described in this application note using Intel's 28F200BX/002BX, proper code segmentation is essential to achieve optimum system performance.

We can divide the system's operating environment into either:

BOOT-TIME CONDITIONS (when system powers-up or is rebooted)

or RUN-TIME CONDITIONS (after boot-up is complete)

The code modules described above are used at different times during normal system operation. Hence, a segregation of the code stored in the flash memory is necessary for proper system operation.

BOOT-TIME EXECUTION

The system requires the boot kernel to be present at the FFFFFH segment during this portion of the cycle.

When using the 28F200BX/002BX flash memory, the first page present will be the top 128 KByte. The rest of the first page is used to store APM code, VGA BIOS code or ExCA socket services. These code modules are then copied from their location below the boot block to system RAM for later initialization after the standard BIOS has started.

Once all code modules are copied into RAM, this first 128 KByte page of flash can be swapped out or exchanged with the 2nd 128 KByte page which is required for run-time execution.

RUN-TIME EXECUTION

Standard BIOS code is required to be present from E0000H to FFFFFH segment during this time period to handle any BIOS calls and maintain proper system operation.

Microsoft MS-DOS 5.0 ROM version code is also required to be present so the BIOS can "SCAN" it in as an adapter.

5.1 16 KByte Recovery Code

The 16 KByte recovery code is critical when a BIOS update does not successfully complete. The reasons for this failure can be divided into two categories:

1. System power failure during BIOS update
2. System reset (soft boot) during BIOS update

The processor will execute recovery code out of the 16 KB boot block when either power is restored or when the system boots up after the reset function.

The 16 KB size recovery code allows the software designer to incorporate as many BIOS system checks as possible to implement basic system functionality.

For instance, these checks may include:

Cursor Positioning
Keyboard services
Time of day service
Basic System services

5.2 28F200BX Reprogramming

Three basic algorithms allow the system designer to reprogram the Flash BIOS chip and perform the necessary tasks for a BIOS update. These algorithms are:

Automated Byte/Word-wide programming
Automated Block erase
Erase Suspend and Resume

For a description of these algorithms, the reader is encouraged to study reference [8]. Appendix B in this application note includes the four flowcharts associated with the algorithms.

To obtain the software drivers necessary to control the device reprogramming operations, consult your local Intel sales office.

5.3 Power Management

Power management is an essential part of any true portable PC design. The design of sophisticated power management techniques is becoming a key differentiator between different machines, and hence is a competitive advantage for the system integrator.

To help the system designer with this often difficult task of optimizing system performance and battery life, the 28F200BX family of products includes three distinct low power modes of operation. These are:

- Standby Current Mode, where the device typically consumes 50 μA

- Automatic Power Savings Feature, where the device typically consumes 1 mA
- Deep powerdown Mode, where the device typically consumes 0.2 μA

For a detailed description of these modes of operation, consult reference [8].

6.0 DESIGNING A 3.3V SYSTEM

The ability to design 3.3V systems used to be a future consideration. Longer battery life and lighter weight portable computers are some of the key objectives for any portable design. Now, thanks to the increasing availability of low voltage components, true low power machines are possible to realize in practice.

6.1 Low Voltage Chips

The list of 3.3V components necessary to build the essential parts of a portable computer is becoming longer every day. Semiconductor manufacturers have recognized the urgent need to supply low voltage chips to the portable marketplace.

The Intel 28F200BX/002BX Boot Block Flash Memories are now sampling in 3.3V versions. These low voltage versions of the 28F200BX/002BX are functionally equivalent to their 5V counterparts and are 100% pinout compatible. So a system converting to 3.3V operation can substitute low voltage 2 Mbit chips (28F200BX-L/002BX-L) when desired without any circuit board modification.

6.2 Power Savings and Improved Battery Life

The 28F200BX/002BX 3.3V chips reduce the total power drawn during normal read operation to less than 25% of the total power in 5V mode. This is a substantial savings in current which translates to 25% less battery drain and hence longer battery life.

Similarly, low voltage version of the most popular microprocessors reduce the total power dissipated by a substantial amount.

The combination of these current savings plus the other system components current reductions improve battery life dramatically and allow the mobile PC user to benefit from weight reduction, longer operating time, lower system cost and higher performance.

7.0 CONCLUSIONS

7.1 Benefits of Extended Flash BIOS

This application note deals with the concepts of extended BIOS implementations in portable PC designs, but it can also be easily adapted to desktop PC systems for which BIOS code requirements can easily exceed 128 KBytes.

We have attempted to explain the requirements and the needs of today's advanced portable BIOS designs which have to meet many difficult and often conflicting requirements.

Boot Block Flash memory is the ideal storage solution to implement the above mentioned features. Furthermore, as the cost of solid state non-volatile flash memory keeps decreasing, the need to switch to these types of media for storing extended BIOS, operating system software, utilities, and in the future application code, becomes more evident and perhaps the only way a PC manufacturer can effectively compete by producing the best engineered, most optimally designed notebooks, palmtop PCs and pen-based computers.

REFERENCES

For more information on the concepts presented in this application note, the reader is encouraged to reference the following documents.

[1] Technical Paper: "Flash: The Optimum BIOS Storage Device" by Brian Dipert, 1991 SVPC — Order Number 297003

[2] "ROM BIOS: The best place for portable PC Power-Management features" by Lance Hansche, 1991 SVPC

[3] AP-341: "Designing an Updatable BIOS Using Flash Memory"—Order Number 292077

[4] AP-357: "Power Supply Solutions for Flash Memory,"—Order Number 292092

[5] Intel386SL Microprocessor Superset System Design Guide—Order Number 240816

[6] Intel386SL Microprocessor Superset Programmer's Reference Manual—Order Number 240815

[7] Intel386SL Microprocessor Superset Data Book—Order Number 240814

[8] Intel 28F200BX/002BX Datasheet—Order Number 290448

[9] Intel 28F400BX/004BX Datasheet—Order Number 290451

[10] Intel 28F200BX-L/002BX-L Datasheet—290449

[11] Intel 28F400BX-L/004BX-L Datasheet—290450

APPENDIX A

Example of a Flash Update Utility Pseudo-Code

This example is for a standard BIOS code and APM code update using the 28F200BX/002BX flash device in an Intel 386SL-based portable computer. Modify this utility, if required, to suit your particular system needs.

Initialize system (set-up user screen, check battery power, check device ID)

Get BIOS/APM file Options (from floppy or through modem)

  If no file present
  Send error message to insert BIOS update floppy, or press ESC to exit

Display BIOS/APM update files, prompt user for choice and load to memory

  If file is invalid
  Prompt for correct file or exit

Inform user of upcoming event, provide option to continue or exit

  If user continues, inform user not to turn off the power or soft-reboot system (CNTL-ALT-DEL)

Erase 28F200BX/002BX device 96 KByte main block

  If system interrupt occurs
  Suspend erase operation if flash memory access is required

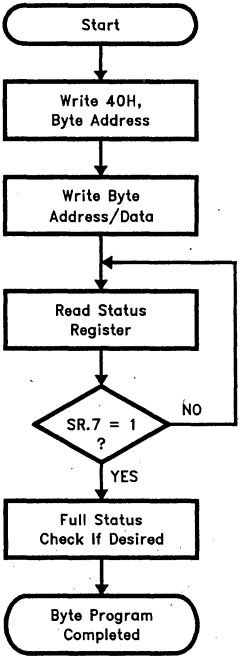
Resume erase operation of 96 KByte main block

Write new file(s) into flash memory 96 KByte main block

Indicate to user that flash reprogramming is complete

Prompt user to reboot the system to continue normal operation

APPENDIX B
28F200BX-T/28F002BX-T Programming Flowcharts



292098-13

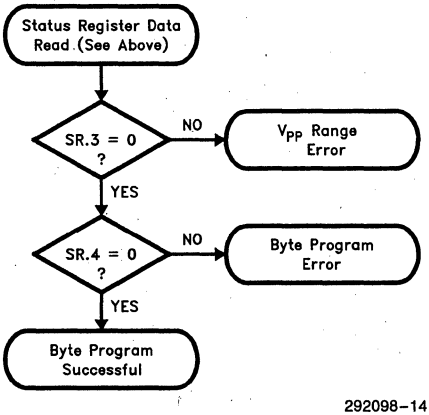
Bus Operation	Command	Comments
Write	Setup Program	Data = 40H Address = Byte to be programmed
Write	Program	Data to be programmed Address = Byte to be programmed
Read		Status Register Data. Toggle $\overline{OE}$ or $\overline{CE}$ to update Status Register
Standby		Check SR.7 1 = Ready, 0 = Busy

Repeat for subsequent bytes.

Full status check can be done after each byte or after a sequence of bytes.

Write FFH after the last byte programming operation to reset the device to Read Array Mode.

Full Status Check Procedure



292098-14

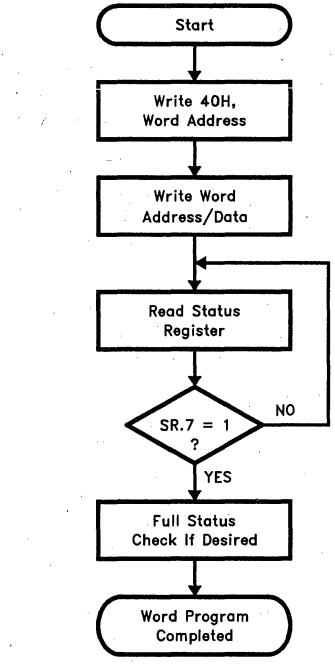
Bus Operation	Command	Comments
Standby		Check SR.3 1 = V_{PP} Low Detect
Standby		Check SR.4 1 = Byte Program Error

SR.3 MUST be cleared, if set during a program attempt, before further attempts are allowed by the Write State Machine.

SR.4 is only cleared by the Clear Status Register Command, in cases where multiple bytes are programmed before full status is checked.

If error is detected, clear the Status Register before attempting retry or other error recovery.

Automated Byte Programming Flowchart



292098-15

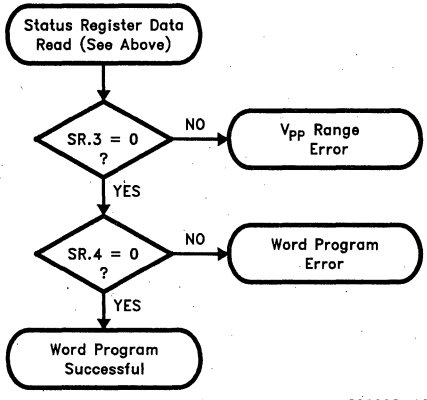
Bus Operation	Command	Comments
Write	Setup Program	Data = 40H Address = Word to be programmed
Write	Program	Data to be programmed Address = Word to be programmed
Read		Status Register Data. Toggle $\overline{OE}$ or $\overline{CE}$ to update Status Register
Standby		Check SR.7 1 = Ready, 0 = Busy

Repeat for subsequent words.

Full status check can be done after each word or after a sequence of words.

Write FFH after the last word programming operation to reset the device to Read Array Mode.

Full Status Check Procedure



292098-16

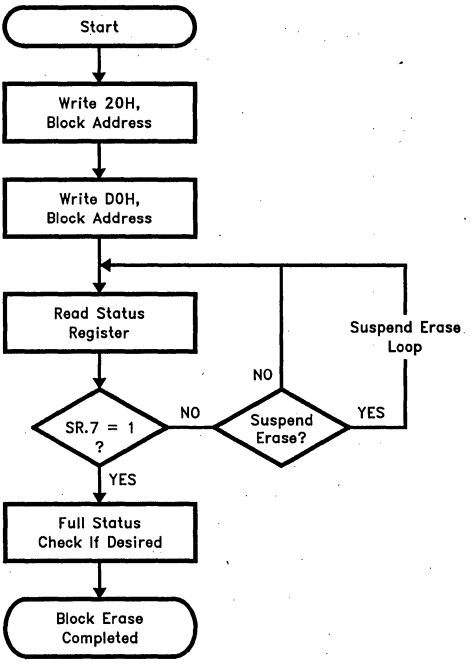
Bus Operation	Command	Comments
Standby		Check SR.3 1 = V_{PP} Low Detect
Standby		Check SR.4 1 = Word Program Error

SR.3 MUST be cleared, if set during a program attempt, before further attempts are allowed by the Write State Machine.

SR.4 is only cleared by the Clear Status Register Command, in cases where multiple words are programmed before full status is checked.

If error is detected, clear the Status Register before attempting retry or other error recovery.

Automated Word Programming Flowchart

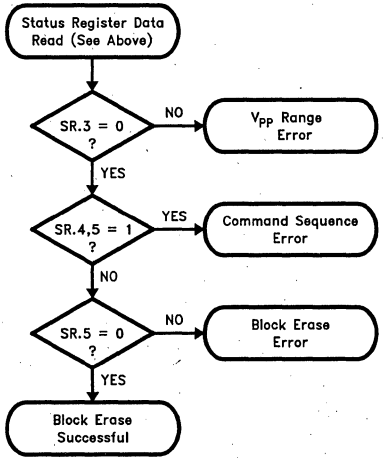


292098-17

Bus Operation	Command	Comments
Write	Setup Erase	Data = 20H Address = Within block to be erased
Write	Erase	Data = D0H Address = Within block to be erased
Read		Status Register Data. Toggle $\overline{OE}$ or $\overline{CE}$ to update Status Register
Standby		Check SR.7 1 = Ready, 0 = Busy
Repeat for subsequent blocks. Full status check can be done after each block or after a sequence of blocks. Write FFH after the last block erase operation to reset the device to Read Array Mode.		

3

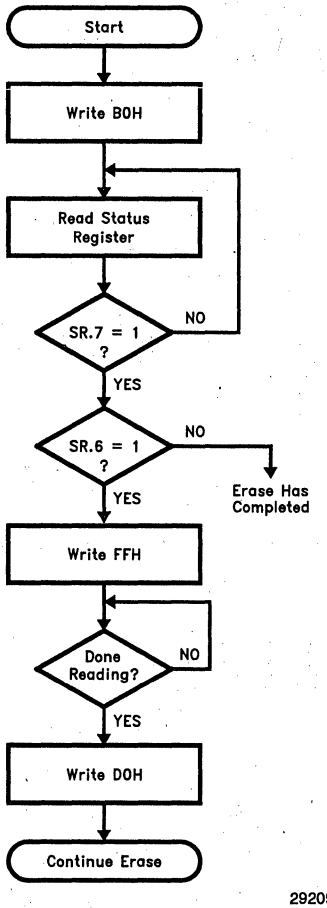
Full Status Check Procedure



292098-18

Bus Operation	Command	Comments
Standby		Check SR.3 1 = V_{PP} Low Detect
Standby		Check SR.4,5 Both 1 = Command Sequence Error
Standby		Check SR.5 1 = Block Erase Error
SR.3 MUST be cleared, if set during an erase attempt, before further attempts are allowed by the Write State Machine. SR.5 is only cleared by the Clear Status Register Command, in cases where multiple blocks are erased before full status is checked. If error is detected, clear the Status Register before attempting retry or other error recovery.		

Automated Block Erase Flowchart



Bus Operation	Command	Comments
Write	Erase Suspend	Data = B0H
Read		Status Register Data. Toggle $\overline{OE}$ or $\overline{CE}$ to update Status Register
Standby		Check SR.7 1 = Ready
Standby		Check SR.6 1 = Suspended
Write	Read Array	Data = FFH
Read		Read array data from block other than that being erased.
Write	Erase Resume	Data = D0H

292098-19

Erase Suspend/Resume Flowchart

APPENDIX C

ROM Chip-Select Decoding Table

COMPLETE ROM DECODINGS

The following table illustrates the decoding of $\overline{\text{ROMCS0}}$ and $\overline{\text{ROMCS1}}$ (not including the FLASH Disk interface decodings):

ROM Chip-Select Decode Table

EBANK2/$\overline{1}$	256KEN	VGAHI	VGALOW	E0DIS	$\overline{\text{ROMCS0}}$	$\overline{\text{ROMCS1}}$
0	0	0	0	0	00E0000H–00FFFFFH 0FE0000H–0FFFFFFH 1FE0000H–1FFFFFFH	
0	0	0	0	1	00F0000H–00FFFFFH 0FE0000H–0FFFFFFH 1FE0000H–1FFFFFFH	
0	0	0	1	0	00C0000H–00C3FFFH 00E0000H–00FFFFFH 0FE0000H–0FFFFFFH 1FE0000H–1FFFFFFH	
0	0	0	1	1	00C0000H–00C3FFFH 00F0000H–00FFFFFH 0FE0000H–0FFFFFFH 1FE0000H–1FFFFFFH	
0	0	1	0	0	00C4000H–00C7FFFH 00E0000H–00FFFFFH 0FE0000H–0FFFFFFH 1FE0000H–1FFFFFFH	
0	0	1	0	1	00C4000H–00C7FFFH 00F0000H–00FFFFFH 0FE0000H–0FFFFFFH 1FE0000H–1FFFFFFH	
0	0	1	1	0	00C0000H–00C7FFFH 00E0000H–00FFFFFH 0FE0000H–0FFFFFFH 1FE0000H–1FFFFFFH	
0	0	1	1	1	00C0000H–00C7FFFH 00F0000H–00FFFFFH 0FE0000H–0FFFFFFH 1FE0000H–1FFFFFFH	
0	1	0	0	0	0FC0000H–0FDFFFFH 1FC0000H–1FDFFFFH 00E0000H–00FFFFFH 0FE0000H–0FFFFFFH 1FE0000H–1FFFFFFH	

ROM Chip-Select Decode Table (Continued)

EBANK2/$\overline{1}$	256KEN	VGAHI	VGALOW	E0DIS	$\overline{\text{ROMCS0}}$	$\overline{\text{ROMCS1}}$
0	1	0	0	1	0FC0000H–0FDFFFFH 1FC0000H–1FDFFFFH 00F0000H–00FFFFFH 0FE0000H–0FFFFFFH 1FE0000H–1FFFFFFH	
0	1	0	1	0	00C0000H–00C3FFFH 0FC0000H–0FDFFFFH 1FC0000H–1FDFFFFH 00E0000H–00FFFFFH 0FE0000H–0FFFFFFH 1FE0000H–1FFFFFFH	
0	1	0	1	1	00C0000H–00C3FFFH 0FC0000H–0FDFFFFH 1FC0000H–1FDFFFFH 00F0000H–00FFFFFH 0FE0000H–0FFFFFFH 1FE0000H–1FFFFFFH	
0	1	1	0	0	00C4000H–00C7FFFH 0FC0000H–0FDFFFFH 1FC0000H–1FDFFFFH 00E0000H–00FFFFFH 0FE0000H–0FFFFFFH 1FE0000H–1FFFFFFH	
0	1	1	0	1	00C4000H–00C7FFFH 0FC0000H–0FDFFFFH 1FC0000H–1FDFFFFH 00F0000H–00FFFFFH 0FE0000H–0FFFFFFH 1FE0000H–1FFFFFFH	
0	1	1	1	0	00C0000H–00C7FFFH 0FC0000H–0FDFFFFH 1FC0000H–1FDFFFFH 00E0000H–00FFFFFH 0FE0000H–0FFFFFFH 1FE0000H–1FFFFFFH	
0	1	1	1	1	00C0000H–00C7FFFH 0FC0000H–0FDFFFFH 1FC0000H–1FDFFFFH 00F0000H–00FFFFFH 0FE0000H–0FFFFFFH 1FE0000H–1FFFFFFH	
1	0	0	0	0	00F0000H–00FFFFFH 0FF0000H–0FFFFFFH 1FF0000H–1FFFFFFH	00E0000H–00EFFFFH 0FE0000H–0FEFFFFH 1FE0000H–1FEFFFFH

ROM Chip-Select Decode Table (Continued)

EBANK2/$\overline{1}$	256KEN	VGAHI	VGALOW	E0DIS	$\overline{\text{ROMCS0}}$	$\overline{\text{ROMCS1}}$
1	0	0	0	1	00F0000H–00FFFFFH 0FF0000H–0FFFFFFH 1FF0000H–1FFFFFFH	0FE0000H–0FEFFFFH 1FE0000H–1FEFFFFH
1	0	0	1	0	00F0000H–00FFFFFH 0FF0000H–0FFFFFFH 1FF0000H–1FFFFFFH	00C0000H–00C3FFFH 00E0000H–00EFFFFH 0FE0000H–0FEFFFFH 1FE0000H–1FEFFFFH
1	0	0	1	1	00F0000H–00FFFFFH 0FF0000H–0FFFFFFH 1FF0000H–1FFFFFFH	00C0000H–00C3FFFH 0FE0000H–0FEFFFFH 1FE0000H–1FEFFFFH
1	0	1	0	0	00F0000H–00FFFFFH 0FF0000H–0FFFFFFH 1FF0000H–1FFFFFFH	00C4000H–00C7FFFH 00E0000H–00EFFFFH 0FE0000H–0FEFFFFH 1FE0000H–1FEFFFFH
1	0	1	0	1	00F0000H–00FFFFFH 0FF0000H–0FFFFFFH 1FF0000H–1FFFFFFH	00C4000H–00C7FFFH 0FE0000H–0FEFFFFH 1FE0000H–1FEFFFFH
1	0	1	1	0	00F0000H–00FFFFFH 0FF0000H–0FFFFFFH 1FF0000H–1FFFFFFH	00C0000H–00C7FFFH 00E0000H–00EFFFFH 0FE0000H–0FEFFFFH 1FE0000H–1FEFFFFH
1	0	1	1	1	00F0000H–00FFFFFH 0FF0000H–0FFFFFFH 1FF0000H–1FFFFFFH	00C0000H–00C7FFFH 0FE0000H–0FEFFFFH 1FE0000H–1FEFFFFH
1	1	0	0	0	00E0000H–00FFFFFH 0FE0000H–0FFFFFFH 1FE0000H–1FFFFFFH	0FC0000H–0FDFFFFH 1FC0000H–1FDFFFFH
1	1	0	0	1	00F0000H–00FFFFFH 0FE0000H–0FFFFFFH 1FE0000H–1FFFFFFH	0FC0000H–0FDFFFFH 1FC0000H–1FDFFFFH
1	1	0	1	0	00C0000H–00C3FFFH 00E0000H–00FFFFFH 0FE0000H–0FFFFFFH 1FE0000H–1FFFFFFH	0FC0000H–0FDFFFFH 1FC0000H–1FDFFFFH
1	1	0	1	1	00C0000H–00C3FFFH 00F0000H–00FFFFFH 0FE0000H–0FFFFFFH 1FE0000H–1FFFFFFH	0FC0000H–0FDFFFFH 1FC0000H–1FDFFFFH
1	1	1	0	0	00C4000H–00C7FFFH 00E0000H–00FFFFFH 0FE0000H–0FFFFFFH 1FE0000H–1FFFFFFH	0FC0000H–0FDFFFFH 1FC0000H–1FDFFFFH
1	1	1	0	1	00C4000H–00C7FFFH 00F0000H–00FFFFFH 0FE0000H–0FFFFFFH 1FE0000H–1FFFFFFH	0FC0000H–0FDFFFFH 1FC0000H–1FDFFFFH

3

ROM Chip-Select Decode Table (Continued)

EBANK2/$\overline{1}$	256KEN	VGAHI	VGALOW	E0DIS	$\overline{\text{ROMCS0}}$	$\overline{\text{ROMCS1}}$
1	1	1	1	0	00C0000H–00C7FFFH 00E0000H–00FFFFFH 0FE0000H–0FFFFFFH 1FE0000H–1FFFFFFH	0FC0000H–0FDFFFFH 1FC0000H–1FDFFFFH
1	1	1	1	1	00C0000H–00C7FFFH 00F0000H–00FFFFFH 0FE0000H–0FFFFFFH 1FE0000H–1FFFFFFH	0FC0000H–0FDFFFFH 1FC0000H–1FDFFFFH

APPENDIX D

List of BIOS software vendors already supporting or announcing their future support for the 28F200BX/002BX flash BIOS chips:

SystemSoft Corporation
Contact: Cliff Sharin
508-651-0088
313 Speen Street
Natick, MA 01760

Phoenix Technologies Ltd.
Contact: Howard Cohen
408-452-6529
40 Airport Parkway
San Jose, CA 95110

Award Software, Inc.
Contact: Jeffry Flink
408-370-7979 (ext. 214)
130 Knowles Drive
Los Gatos, CA 95030-1832

American Megatrends, Incorporated
Contact: Tom Rau
404-246-8612
1346 Oakbrook Drive, Suite 120
Norcross, GA 30093

Quadtel
Contact: Dale Buscaino
714-754-4422 (ext. 250)
3190-J Airport Loop
Costa Mesa, CA 92626